



 Pay Day Sale is Now Live!  Use the Coupon **PAYDAY** and Get Maximum Discount [Enrol Now!](#)



 Java Programming Handbook



[Handbooks](#) > [Java Programming Handbook](#) > [Java if else statement](#)

Java If-Else Statement

The **if-else** statement is one of the most fundamental decision-making structures in programming. It allows your program to execute different code based on whether a condition is true or false. In Java, we use the **if-else** statement to control the flow of execution depending on certain conditions.

In this blog, we'll explore how the **if-else** statement works in Java, provide explanations, and demonstrate examples with expected outputs.

What is an If-Else Statement?

An **if-else** statement is used to execute one block of code if a condition is true, and another block if the condition is false. It's a way to introduce logic in your program, so it can make decisions.

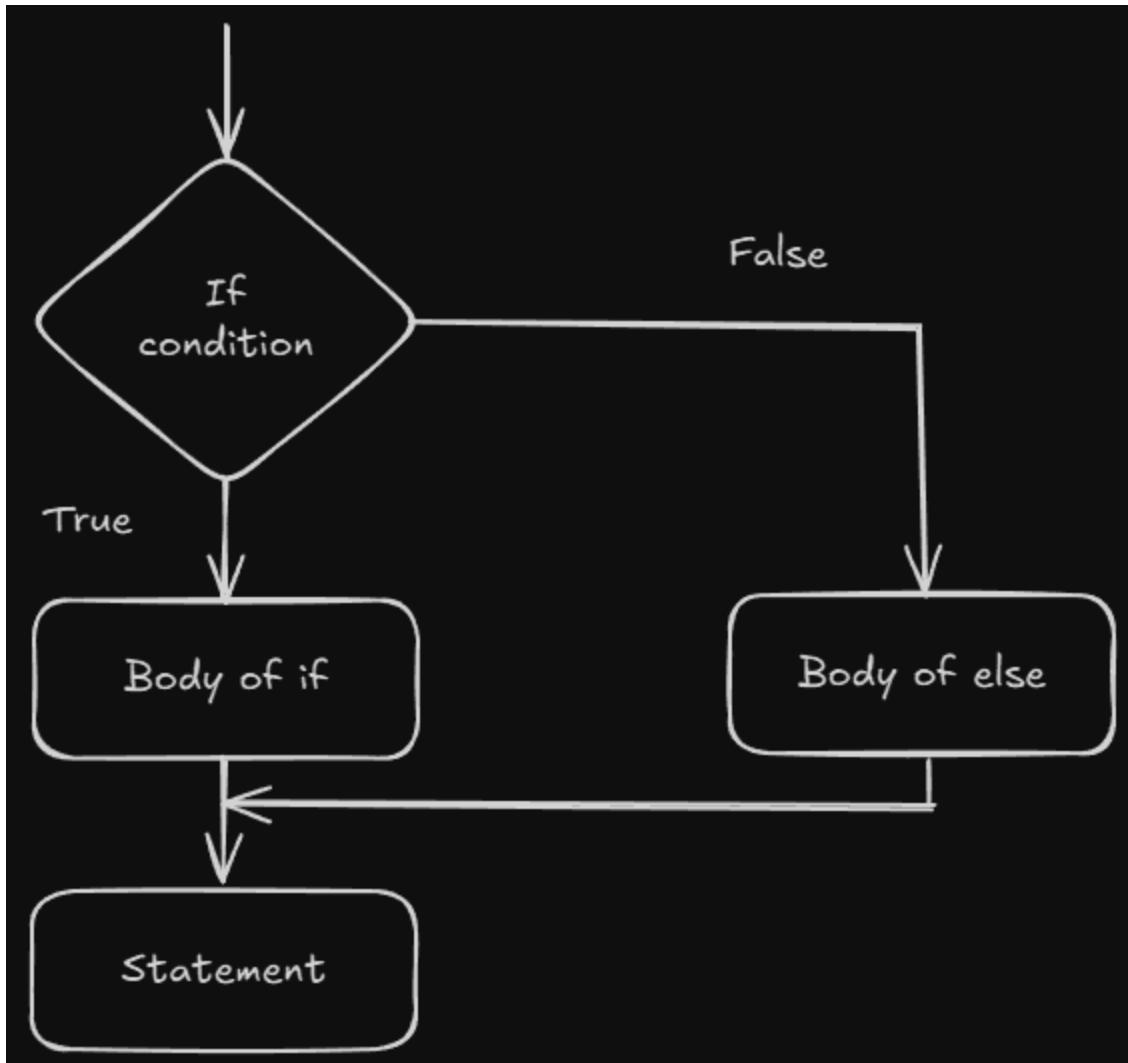
The syntax for an if-else statement in Java is:



```
if (condition) {  
    // code to be executed if the condition is true  
} else {  
    // code to be executed if the condition is false  
}
```

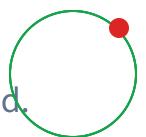


Flow of Execution



If else flow chart

1. **If condition:** The condition inside the `if` is evaluated. If the condition is `true`, the block of code inside the `if` is executed.
2. **Else block:** If the condition is `false`, the block of code inside the `else` is executed.



Basic Example: If-Else Statement

Let's start with a simple example to see how the **if-else** statement works.

Example 1: Basic If-Else Statement

```
class Main {  
    public static void main(String[] args) {  
        int number = 10;  
  
        // If-Else statement to check if number is positive or negative  
        if (number > 0) {  
            System.out.println("The number is positive.");  
        } else {  
            System.out.println("The number is negative.");  
        }  
    }  
}
```

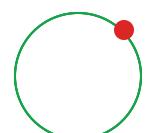
Explanation:

- The condition `number > 0` is evaluated. Since the value of `number` is 10, which is greater than 0, the condition is `true`, and the code inside the `if` block is executed.
- If the number was negative or zero, the code inside the `else` block would have been executed.

Expected Output:

```
The number is positive.
```

If-Else with Multiple Conditions: Using `else if`



Sometimes, we want to check for more than two conditions. In this case, we can use the `else if` statement to check additional conditions after the initial `if` condition.

Example 2: If-Else-If Statement

```
class Main {  
    public static void main(String[] args) {  
        int number = 0;  
  
        // If-Else-If statement to check if the number is positive, negative, or  
        if (number > 0) {  
            System.out.println("The number is positive.");  
        } else if (number < 0) {  
            System.out.println("The number is negative.");  
        } else {  
            System.out.println("The number is zero.");  
        }  
    }  
}
```

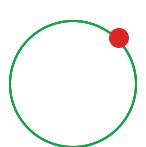
Explanation:

- The first condition checks if the number is positive. Since the number is 0, it moves to the `else if` condition and checks if the number is negative.
- Finally, since the number is neither positive nor negative, the `else` block is executed, and the program prints "The number is zero."

Expected Output:

```
The number is zero.
```

Nested If-Else Statement



A **nested if-else** statement is when you place an **if-else** statement inside another **if** or **else** block. This is useful when you have more complex conditions to check.

Example 3: Nested If-Else Statement

```
class Main {  
    public static void main(String[] args) {  
        int age = 18;  
  
        // Nested if-else to check voting eligibility  
        if (age >= 18) {  
            if (age == 18) {  
                System.out.println("You are 18 years old and eligible to vote for the");  
            } else {  
                System.out.println("You are eligible to vote.");  
            }  
        } else {  
            System.out.println("You are not eligible to vote.");  
        }  
    }  
}
```

Explanation:

- The first **if** checks if the age is greater than or equal to 18.
- Inside the **if** block, there's another **if** that checks if the age is exactly 18.
- If the age is 18, it prints a specific message for first-time voters. Otherwise, it simply prints that the person is eligible to vote.

Expected Output:

You are 18 years old and eligible to vote for the first time.

If-Else with Logical Operators

You can also combine multiple conditions in an `if` statement using logical operators like `&&` (AND), `||` (OR), and `!` (NOT). This helps you to create more complex conditions.

Example 4: If-Else with Logical AND (&&)

```
class Main {  
    public static void main(String[] args) {  
        int age = 25;  
        boolean hasVoterID = true;  
  
        // If-Else with Logical AND (&&) to check voting eligibility  
        if (age >= 18 && hasVoterID) {  
            System.out.println("You are eligible to vote.");  
        } else {  
            System.out.println("You are not eligible to vote.");  
        }  
    }  
}
```

Explanation:

- The condition `age >= 18 && hasVoterID` checks if both conditions are true. Since both are true in this case (age is 25 and the person has a voter ID), the `if` block is executed.

Expected Output:

```
You are eligible to vote.
```

Example 5: If-Else with Logical OR (||)

```
class Main {  
    public static void main(String[] args) {
```

```
int age = 17;
boolean hasParentalConsent = true;

// If-Else with Logical OR (||) to check voting eligibility with consent
if (age >= 18 || hasParentalConsent) {
    System.out.println("You are eligible to vote.");
} else {
    System.out.println("You are not eligible to vote.");
}
}
```

Explanation:

- The condition `age >= 18 || hasParentalConsent` checks if either the age is 18 or older or the person has parental consent.
- Since the person is 17 but has parental consent, the `if` block is executed.

Expected Output:

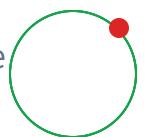
You are eligible to vote.



Conclusion

In this blog, we've learned:

- The `if-else` statement allows you to make decisions in your program by checking conditions.
- The `if-else` structure can handle simple or multiple conditions using `if`, `else if`, and `else`.
- We can also use **logical operators** like `&&` (AND) and `||` (OR) to combine multiple conditions.



- Nested if-else statements allow more complex logic.

Mastering the **if-else** statement is crucial for writing programs that make decisions based on data, and it's a fundamental concept in programming. With these building blocks, you can create powerful decision-making logic in your Java applications.

Want to Master Spring Boot and Land Your Dream Job?

Struggling with coding interviews? Learn Data Structures & Algorithms (DSA) with our expert-led course. Build strong problem-solving skills, write optimized code, and crack top tech interviews with ease

[Learn more](#)

Last updated on Apr 09, 2025

Subscribe to our newsletter

Read articles from Coding Shuttle directly inside your inbox. Subscribe to the newsletter, and don't miss out.

Enter your email address

SUBSCRIBE

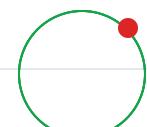
We recommend

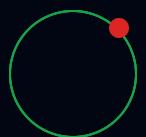


Spring Boot 0 to 100 Cohort 4.0 [AI + DevOps]

₹9,990 ₹27,990

CHECK IT OUT →



[Previous](#)[Next](#)[**<< Java Expressions, Statements and Blocks**](#)[**Java Ternary Operator >>**](#)*#BetterEveryday***FOLLOW US ON****COMPANY**[About Us](#)[Terms & Conditions](#)[Privacy Policy](#)[Pricing & Refund Policy](#)[Contact Us](#)[Our Blogs](#)**RESOURCES**[Handbooks](#)[Mock Tests](#)[DSA Sheets](#)[Blogs](#)[Newsletter](#)**COURSES**[Spring Boot 0 to 100 Cohort 4.0 \[AI + DevOps\]](#)[DSA Prime 4.0](#)[React 19 Course 0 To 100](#)[Java React Full Stack Course 2.0](#)**POPULAR HANDBOOKS**[Spring Boot Handbook](#)[Java Programming Handbook](#)[Kubernetes Handbook](#)**ONLINE COMPILERS**[Online Java Compiler](#)[Online C++ Compiler](#)[Online Python Compiler](#)[Online Javascript Compiler](#)[Online Go Compiler](#)[Online C# Compiler](#)[Online C Compiler](#)**DSA SHEETS**[CS 45 Sheet](#)[CS SDE Sheet](#)[DSA Prime Sheet](#)[Blind 75 Sheet](#)

MOCK TESTS

Quantitative Aptitude for Placements | Logical Aptitude for Placements | C++ OOPS for Interviews |

Java OOPS for Interviews | Javascript for Interviews | Python for Interviews |

C++ Language for Interviews | C Language for Interviews | Java for Interviews |

Operating Systems | SQL for Interviews | RDBMS for Interviews |

Docker & Kubernetes for Interviews | Spring Cloud for Interviews |

Advanced Spring Boot for Interviews | Spring Security for Interviews |

Spring Data JPA for Interviews | Core Spring Boot for Interviews |

Copyright © NGU Education Pvt. Ltd.

